Bespoke Project Design Documentation & Technical Report

Aiden Bond - Whistl Fulfilment South West
Post-Development Version 1 - 02/01/2024

Contents

Bespoke Project - Pre Development

Introduction

The Bespoke Project software will be designed to streamline the logging of bespoke items, particularly when the processing and fulfilment of these items deviate from standard procedures.

Applications & Usage

- A user has a Goods In delivery of bespoke products which are made to order, these need to be logged.
- The Goods In delivery will have a purchase order number or batch number and could consist of multiple boxes.
- The user will be able to use the application to record the purchase order number/batch number, and all of the SKUs/Orders that came in on that PO/BN as well as which box they were in.
- The application will have a parameter for location, this will be the location where the products will be assigned to. This location will be recorded with the rest of the data and is set by a default parameter.

Forms

- **Login Form**
  The form where users will log into the application

- **Main Form**
  The main for the user will use to record the goods in data

- **Search Form**
  The form where users will be able to search for past records that have been recorded

- **Maintenance Form**
  The form where users will be able to edit and amend past records that have been recorded

Database Tables

In order to facilitate data logging, the application relies on three essential SQL tables. These tables are responsible for storing recorded data, default parameters, and settings that the application can utilise. The incorporation of default parameters not only accommodates future expansion to additional clients and stock management processes but also ensures the long-term viability of the application.

BespokeProject_Details

This table will be used to store the data logged by the user when the app is being used. This table can also be used to report on the logged data for future analysis.

Columns:

- [Batch_No] [VARCHAR] (24) - This will be the Batch Number
- [BoxID] [VARCHAR] (24) - This will be the BoxID
- [Ref_No] [VARCHAR] (24) - This will be the order reference
- [Location] [VARCHAR] (6) - This will be the location
- [DT_Created] [DATETIME] - This will be the date
- [User_Created] [VARCHAR] (24) - This will be the user
- [Last_Upd_User] [VARCHAR] (24) - This will be the last updated user
- [Last_Updated] - This will be the date the records were last updated

BespokeProject_Parameters

This table will be used to store default data that the app will use.

Columns:

- [Parameter] [VARCHAR] (100) - This will be the parameter
- [String_01] [VARCHAR] (255) - This can be used to store a default string
- [Date_01] [DATETIME] - This can be used to store a default date
- [Int_01] [int] - This can be used to store a default int

BespokeProject_Locations

This table will be used to store the different locations that can be used within the app.

Columns:

- [Location] [VARCHAR] (6) - This will be PTN & WTN

Stored Procedures

This application will execute stored procedures in order to operate.
The utilisation of stored procedures ensures that most SQL processes are encapsulated within the database, offering a more straightforward approach to maintenance or modifications when issues arise relating to the database components. Consequently, this approach reduces the need for maintenance within the application itself.

BespokeProject_Insert_Details

This stored procedure will be used on the main form and will contain the main SQL Insert query that inserts the data logged by the user into the SQL database
The application will pass through 5 parameters to the stored procedure:

@Batch_No - This will be the batch number logged by the user.
@BoxID - This will be the Box ID logged by the user.
@Ref_No - This will be the reference number logged by the user.
@User - This will be the user that logged into the application.
@Location - This will be the location pulled through from the default parameters table.

BespokeProject_Update_Details

This stored procedure will be used in the Maintenance Form and will be used to execute the SQL Update query that will update the selected records based on the data inputted by the user. The application will pass through 4 parameters to the stored procedure:

@Batch_No - This will be the batch number the users wishes to update records for.
@Field - This will be the field the user wishes to update.
@New_Data - This will be the new data that the selected field will be updated to.
@Selected_Records - This will be a selected records the user wishes to update.

BespokeProject_Get_Details

This stored procedure will be used in the Main Form and will be used to execute an SQL Select query that will populate the history table.
This stored procedure will also be used in the Search Form to gather the data for the search results table.
If the @BoxID and the @Ref_No Parameters are NULL the Stored procedure will pull all data relating to the Batch Number. The application will pass through 3 parameters:

@Batch_No - This will be the batch number logged by the user.
@BoxID - This will be the Box ID logged by the user.
@Ref_No - This will be the reference number logged by the user.

BespokeProject_Delete_Details

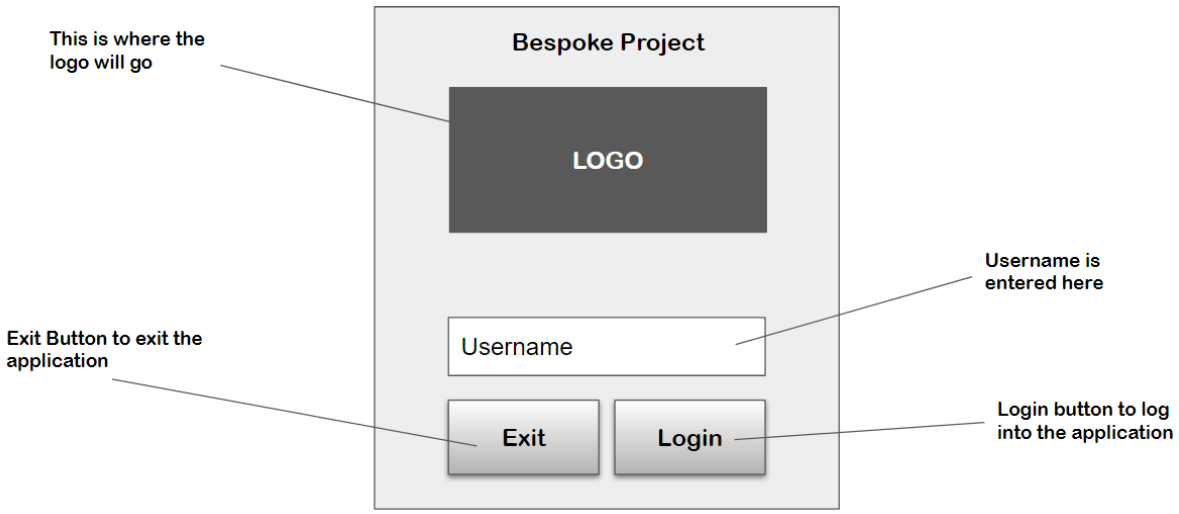
This stored procedure will be used in the Maintenance Form and will be used to execute the SQL Delete query that will delete the selected records that have been selected by the user The application will pass through 2 parameters to the stored procedure:

@Batch_No - This will be the batch number that the user wishes to delete records from.
@Selected_Records - this will be the selected records the user wished to delete.

Forms Design

The application will use a total of 4 forms in order to operate, these forms will allow the user to login to the application, as well as log the required data, search through previously logged data and amend previously logged data.

Login Form



- This form will be where the user logs in, no password required.
- The username will feed through to the details table when data is logged.
- The username has to be a valid Elucid username as requested by the client (Whistl Fulfilment South West).
  Elucid is the name of the Warehouse Management System used by Whistl Fulfilment South West.
- The “Exit” button will ask the user if they want to exit, a message will appear saying “Are you sure you want to Exit?”
- The “Login” button will take the user to the Main Form.
- The logo will be the Whistl company logo or application logo.

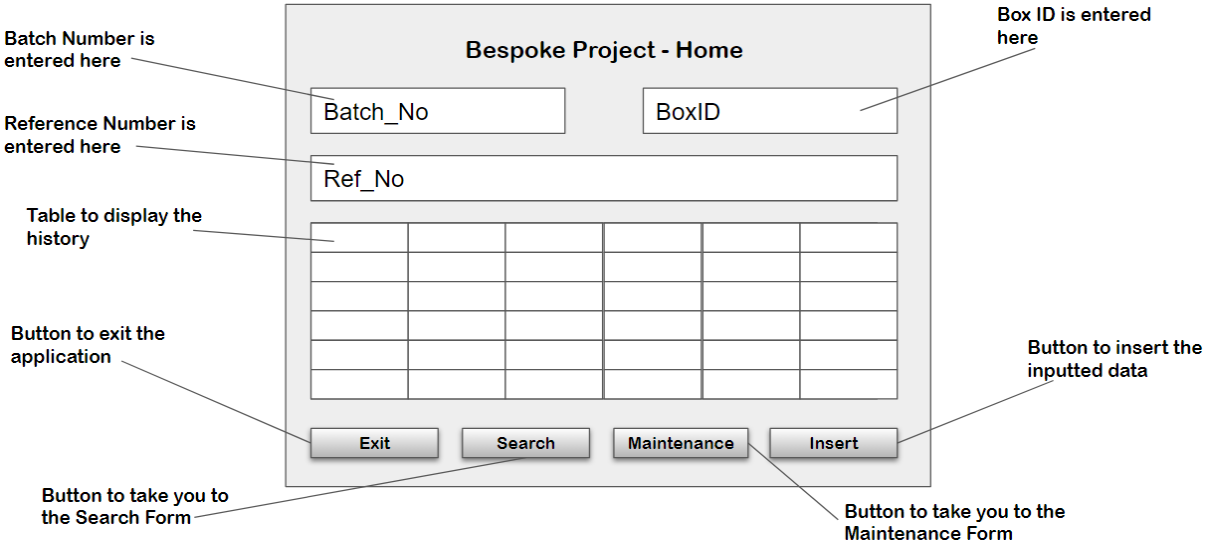
Required Stored Procedures & SQL

- A validity check is required to make sure the username entered is a valid username in the database.

Error Handling

- If a user enters an username that is not valid, a message will appear asking for the user to enter a valid username.

Main Form



- This Form is where the user will be able to enter the data to be logged in the database.
- The user will enter a Batch Number, BoxID & Reference Number and when the Tab Key or the “Insert” button is pressed, the data will be inserted into the database.
- Both the Batch Number & BoxID fields will not automatically clear after the insert event. This will allow users to enter multiple references under the same Batch Number & BoxID.
- When a Batch Number is entered the table will populate with records where the Batch Number is the same as the batch number entered.
- When the batch number is entered and the user navigates to the next field, the table will populate with all the data in the BespokeProject_Details table associated with that batch number.
- As the user inserts records, the table will update with those records.
- The “Exit” button will ask the user if they want to exit, a message will appear saying “Are you sure you want to Exit?”

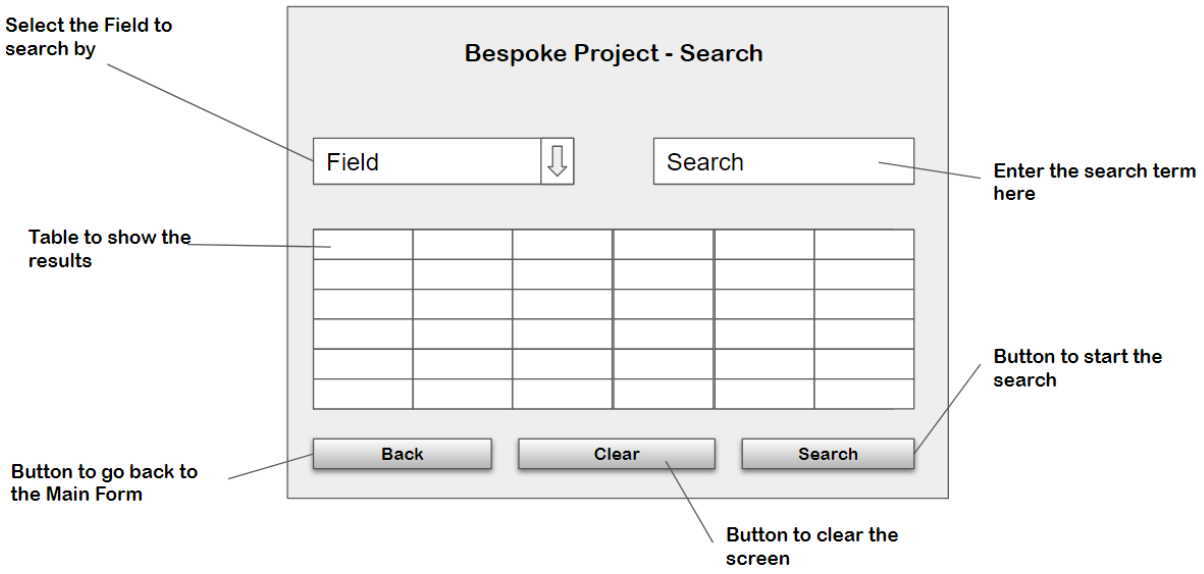
Required Stored Procedures & SQL

- BespokeProject_Insert_Details
- BespokeProject_Get_Details

Error Handling

- If the user has not completed all the fields before clicking the insert button or pressing the tab key, an error will appear asking the user to make sure all fields are completed.
- If the user enters a Reference Number that is already recorded under the Batch Number and Box ID an error will appear informing the user of a duplicate record.

Search Form



- This form will be used to search for records in the BespokeProject_Details table.
- The user will be able to search based on a selected field & reference, the field will be selected from a drop down box.
- The results will be displayed in the table the same as on the Main Form.

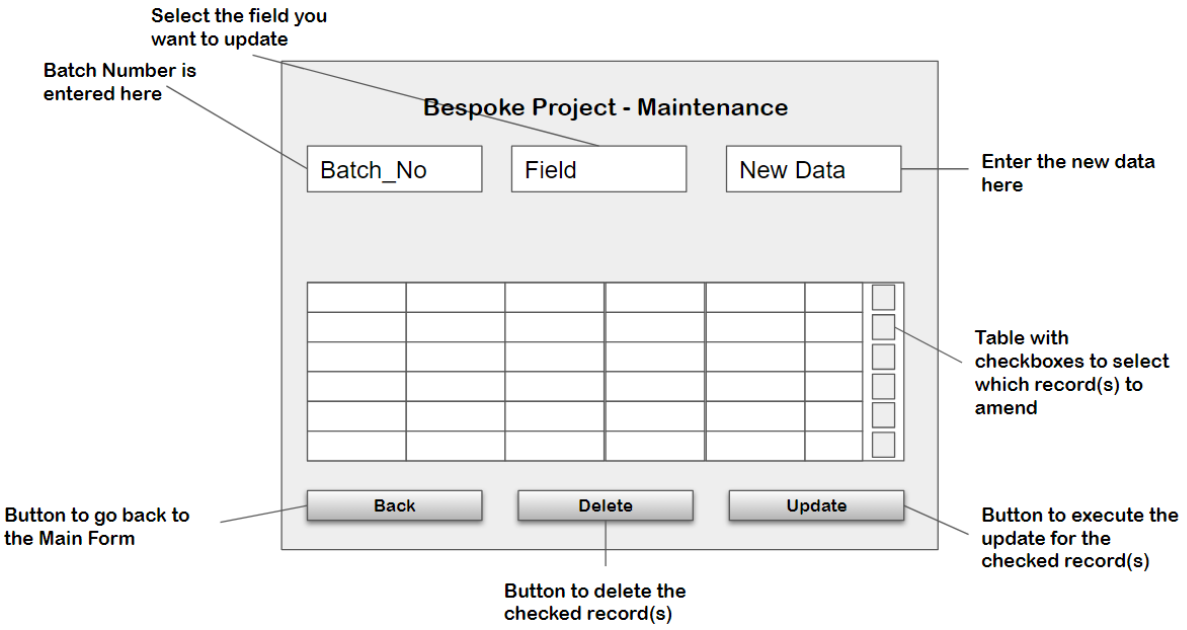
Required Stored Procedures & SQL

- BespokeProject_Get_Details

Error Handling

- If the user does not enter a search term and/or select a field before clicking the search button, an error will appear asking the user to complete the all fields
- If no records are pulled from the search parameters an error will appear advising the user that no records were found

Maintenance Form



- This Form will allow users to amend or delete records in the BespokeProject_Details table.
- Users will enter the Batch Number, check the checkbox for the record(s) they'd like to amend or delete. If they want to delete the record(s) they will click the "Delete" button. If they'd like to amend the record(s) they can select the field they'd like to update & enter the new data, then click the "Update" button.
- When the batch number is entered and the user navigates to the next field, the table will populate with all the data in the BespokeProject_Details table associated with that batch number.

Required Stored Procedures & SQL

- BespokeProject_Update_Details
- BespokeProject_Delete_Details

Error Handling

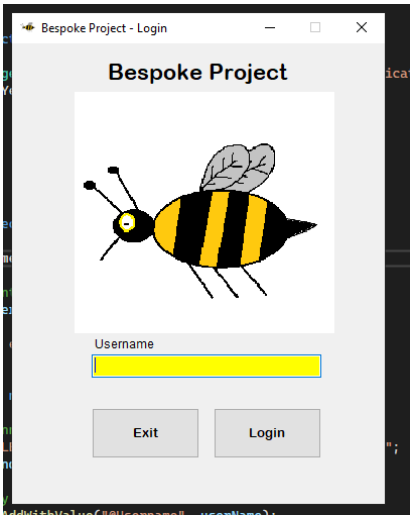
- If the user clicks the update button before completing all the fields and selecting the records they'd like to update, an error will appear asking the user to complete all the fields
- If the user clicks the delete button before completing all the fields and selecting the records they'd like to delete, an error will appear asking the user to complete all the fields

Bespoke Project - Development

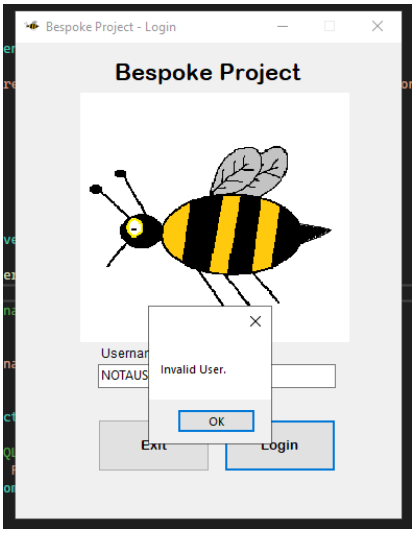
Throughout the development phase, several challenges were overcome in order to get the application to meet the original pre-development brief. This as well as some design and functionality changes along the way have resulted in a working application that meets the criteria specified in the pre-development brief as well as some added “Quality of life” features.
In the course of application development, I aimed to simplify processes by encapsulating them within methods, ensuring code readability, understanding, and ease of editing when necessary. A concerted effort was made to adhere to a coding convention that promotes clarity. I also utilised comments extensively to provide explanations for each line or section of code.

Login Form

The login form allows users to log into the application using their Elucid username. The application checks the database to ensure the user is valid. If they are not, an error will appear; if they are, the application will load the Main Form for the user.



Here we can see the login form with a logo for the application as well as the title, Username Text box (***txbUsername***), Login Button (***btnLogin***) and Exit Button (***btnExit***).

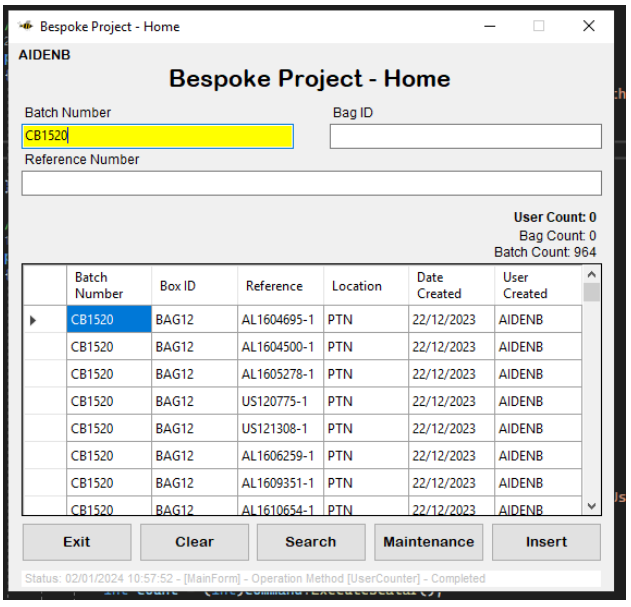


Here we can see an error has occurred because an invalid username has been entered into ***txbUsername*** and ***btnLogin*** has been clicked.

Main Form

The Main Form allows users to input a Batch Number, Bag ID, and Reference Number, inserting these records into the SQL database. During development and testing, a decision was made to relabel the Box ID text box (***txbBoxID***) to 'Bag ID.' This change accommodated the application's testing in a production environment where received stock was packaged in bags rather than boxes. Furthermore, additional features were incorporated into this form to enhance functionality and improve the user experience. Features include:

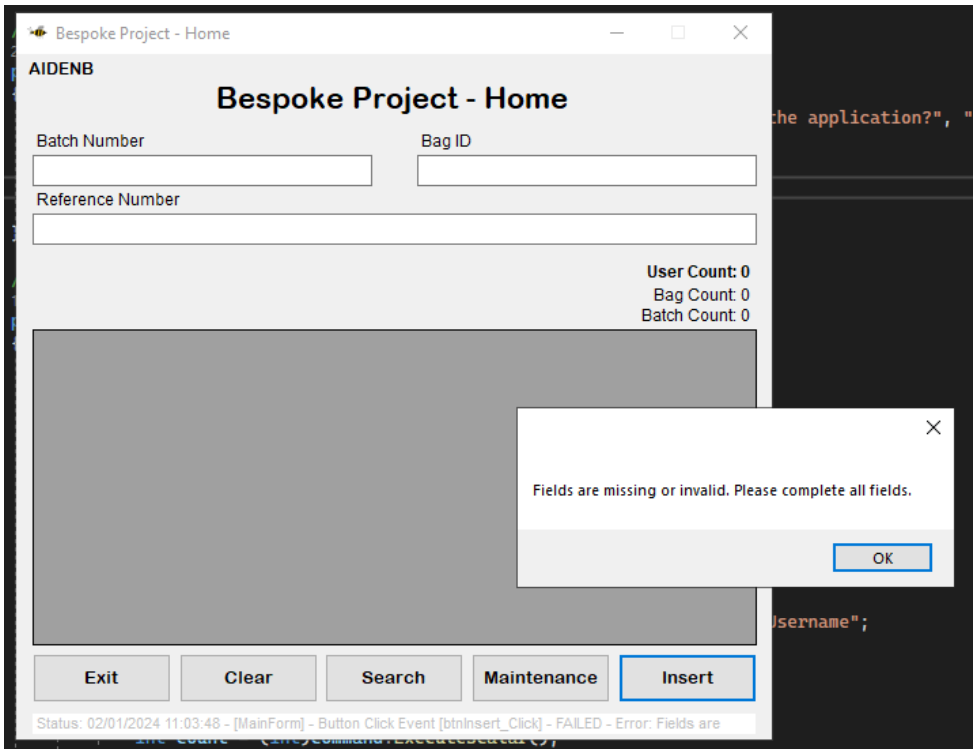
- **Clear Button** (***btnClear***) which will clear all text boxes and the datatable,This functionality allows users to start a new batch by simply clicking the button, facilitating a seamless reset of the interface.
- **Counters** have been implemented to provide indications of the quantities being entered. These counters include a user counter, which displays the total inputs made by the logged-in user for the current day. Additionally, there is a bag counter, indicating the total inputs associated with the bag number entered in ***txbBoxID***, and a Batch Counter, showing the total inputs against the Batch Number in the Batch Number text box (***txbBatchNo***)
- **Status Box/Logging** feature which displays the last operation executed by the application, aiding in debugging by identifying potential issues. Additionally, a hard coded parameter, when set to 'true,' generates a log file during application usage. This file captures all the data displayed in the Status Box each time it updates.
- **Keyboard Shortcuts** implemented to enhance navigation, improve ease of use, and increase accessibility. In accordance with the pre-development brief, the Tab key is configured to trigger the same insert event as the 'Insert' button (***btnInsert***). All text box fields support navigation using Tab and Shift+Tab. Additionally, various forms can be opened by using Ctrl plus the first letter of the form's name (e.g., Search Form = 'Ctrl+S'). The 'Clear' button's event (***btnClear***) can also be triggered by 'Ctrl+G,' and the 'Exit' button (***btnExit***) can be activated using the 'Esc' key."



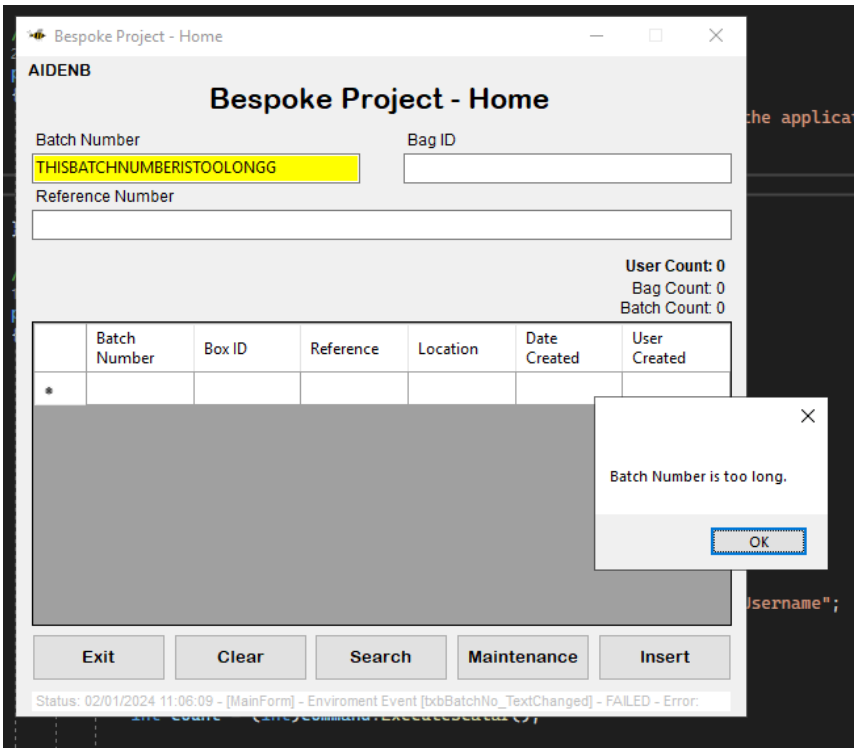
Here we can see the Main Form with the Batch Number text box (***txbBatchNo***), Bag ID text box (***txbBoxID***), and reference Number text box (***txbRefNo***). We can also see the table showing previous inputted data relating to the batch number in ***txbBatchNo*** as well as the addition features such as btnClear, the Counters and the status Box

The Main Form incorporates multiple error-checking and handling events to ensure robust and stable operation. This includes checking the lengths of inputted fields, restricting certain special characters, and implementing a Try/Catch process. This comprehensive approach aims to identify and address potential issues, enhancing the overall reliability of the application.

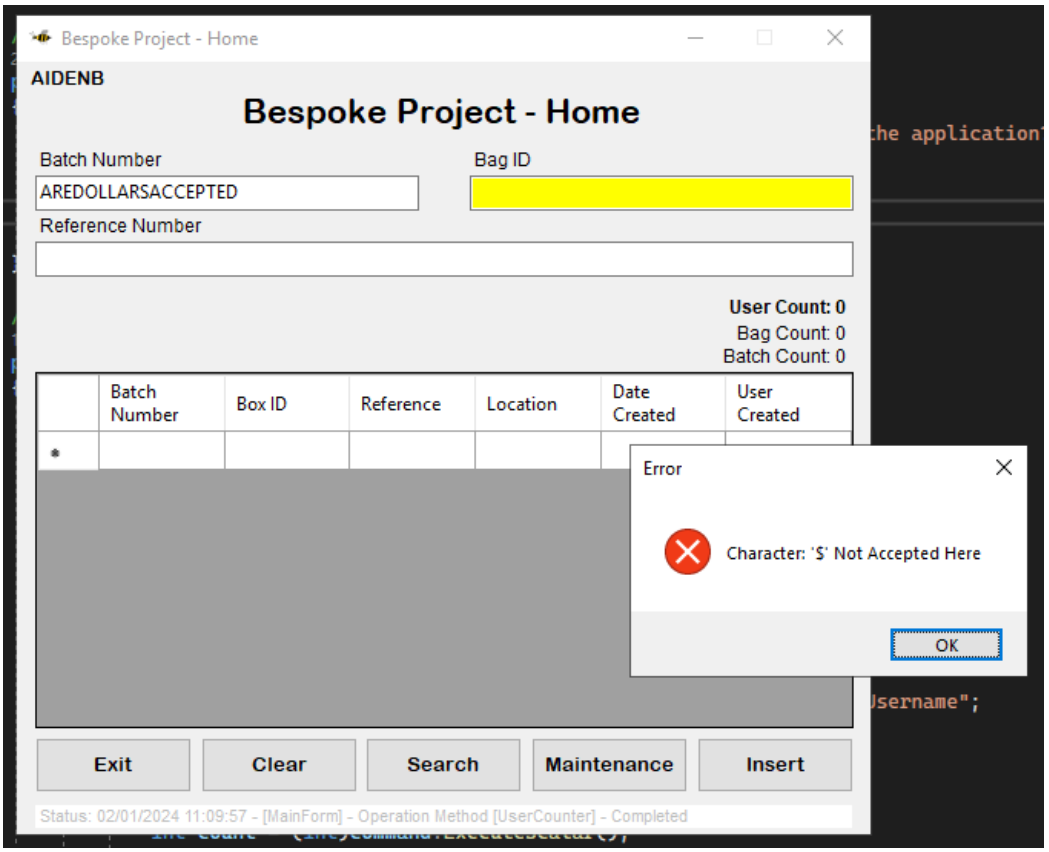
Here we can see the user has attempted to input data without completing one or more of the fields



Here we can see the user has entered a batch number that is too long, an error message has popped up and the field will be highlighted. This repeats for all text boxes



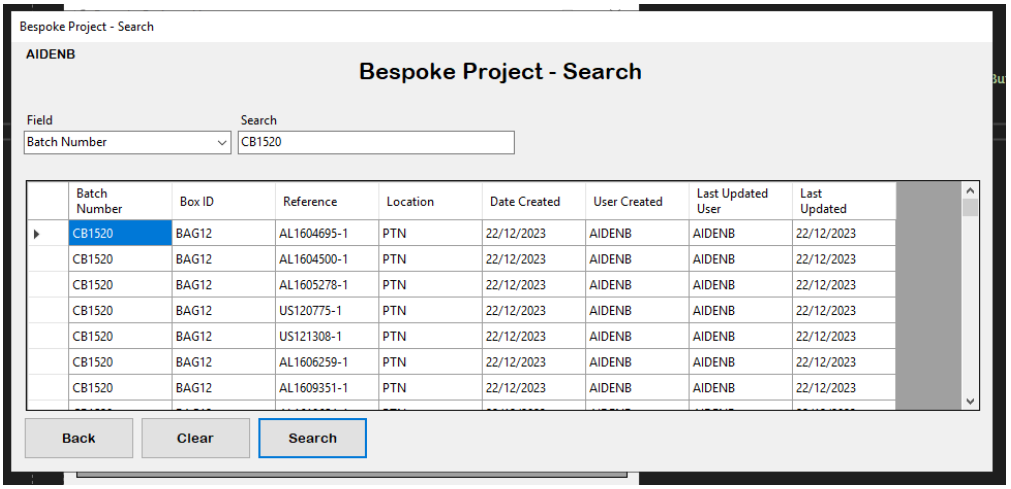
Here we can see the user has attempted to enter a special character into the ***txbBoxID*** this error will occur if you attempt to enter certain special characters into any of the text boxes

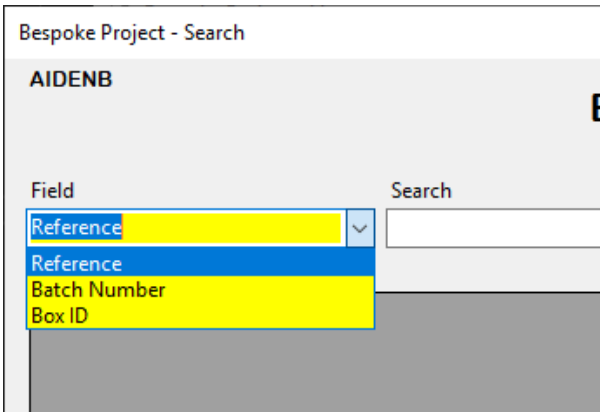


Search Form

The Search Form allows users to search for records in the database based on a search term and a selected field. This form is straightforward and required minimal debugging, as user interactions are limited to entering values, clicking 'Search,' and then either clearing the screen to start a new search or exiting the form

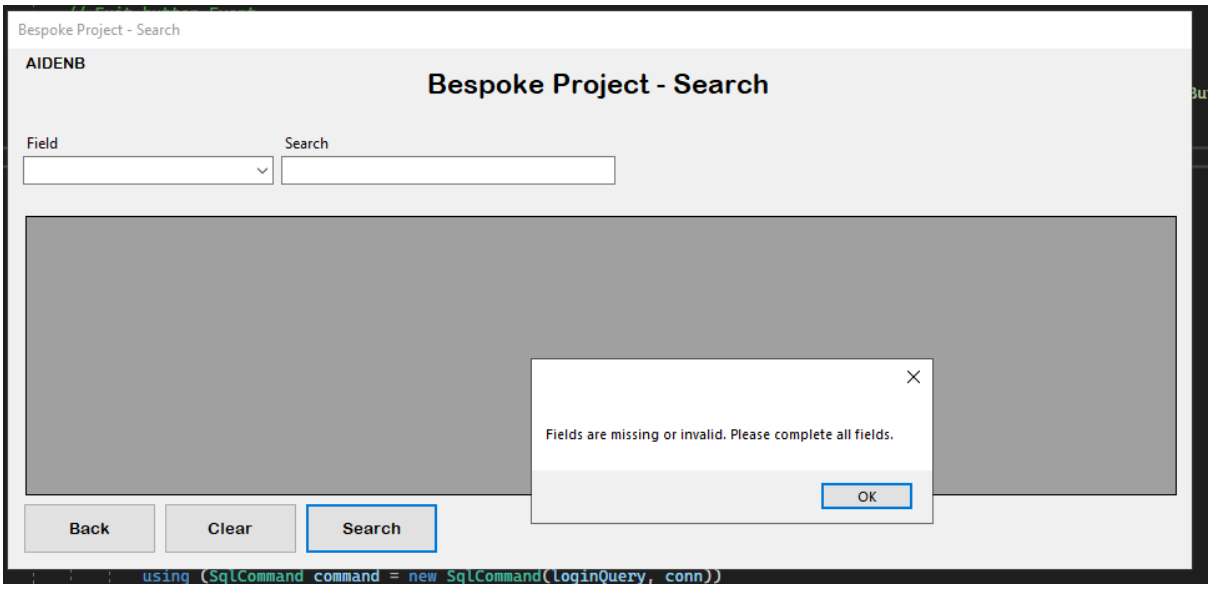
Here we can see the search from with a drop down box to select the field you'd like the search term to relate to and a text box to enter a search term, as well as a table displaying the search results, a button to clear the screen (***btnClear***), a button to start the search (***btnSearch***) and a button to close the form (***btnBack***)



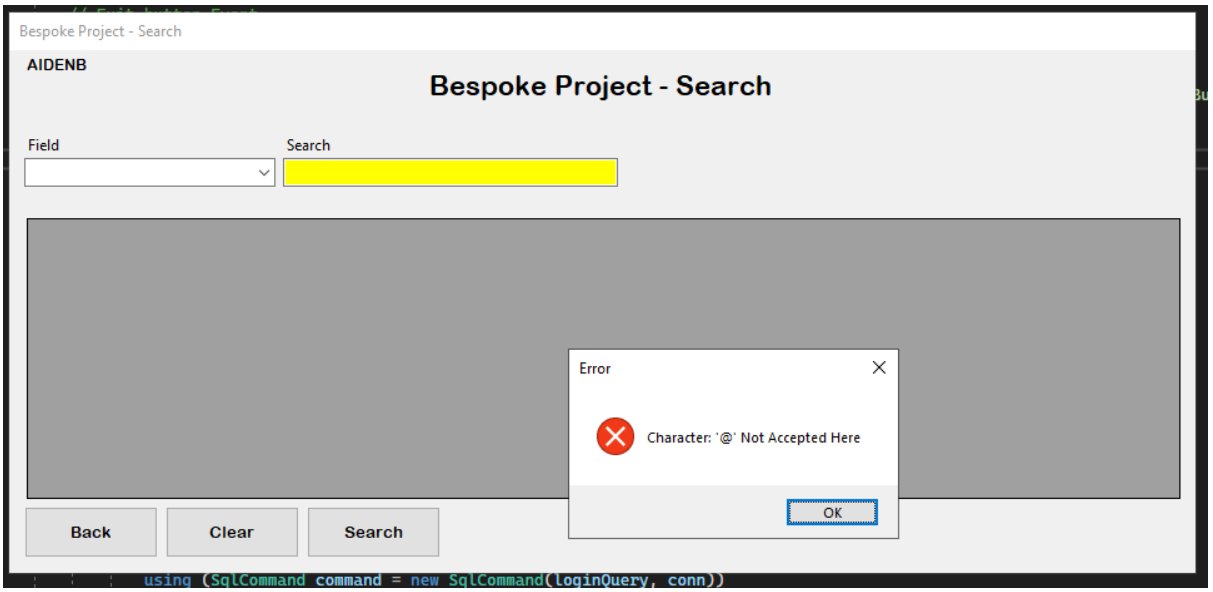


Here we can see the dropdown box with option for the fields you'd like to search by

Here we can see a user has attempted to carry out a search without completing one or more of the fields, this has resulted in an error.



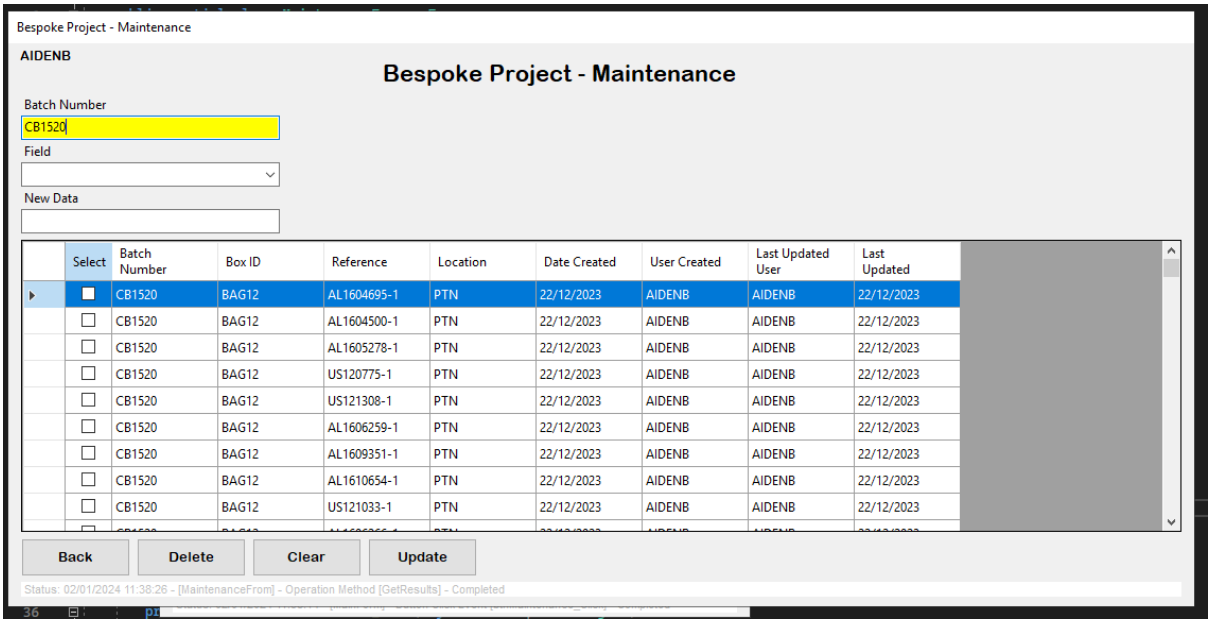
Here we can see the user has attempted to input a special character into ***txbSearch***

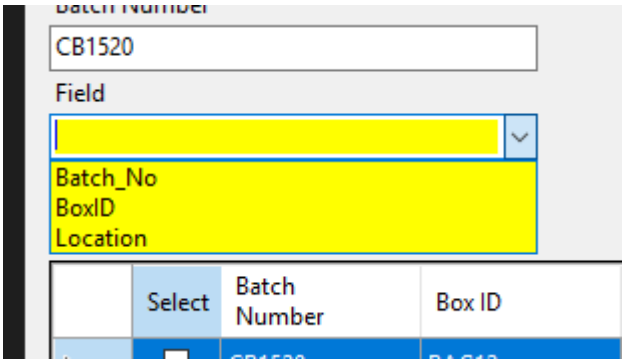


Maintenance Form

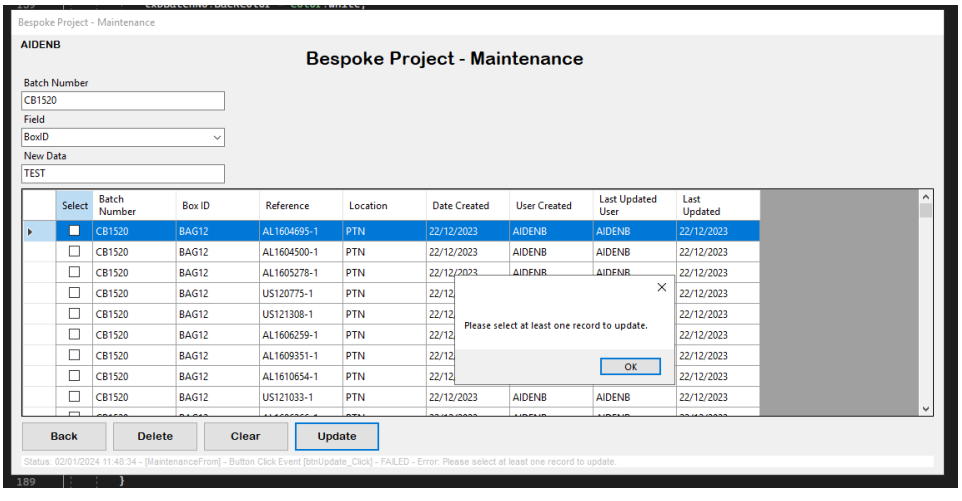
The Maintenance Form allows users to delete inputted record(s) from the database or edit record(s) in the database by changing the values for one or more of their fields. This form necessitates the use of a table that allows user interaction, enabling them to select the specific record(s) they wish to delete or edit.

Here we can see the Maintenance Form with a text box for a Batch Number (***txbBatchNo***), text box for new data to to be inputted (***txbNewData***) and a dropdown box to select the field the users wishes to update (***cbField***) as well as a table showing the records relating to the batch number entered in ***txbBatchNo***, each row has a text box for the user to select



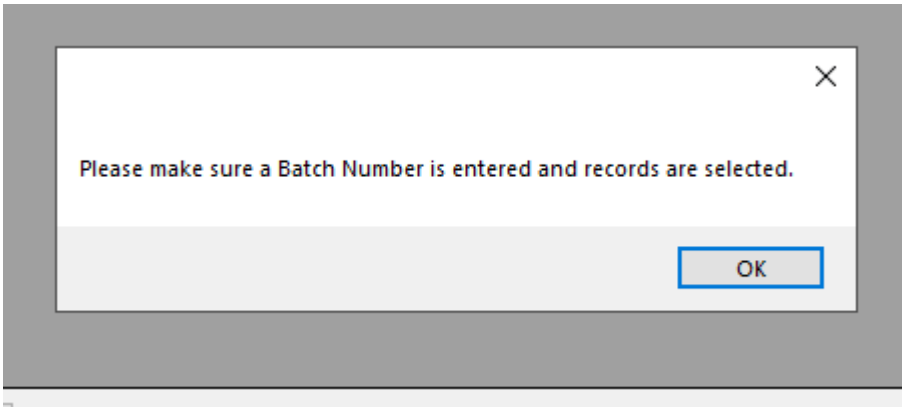


Here we can see the drop down box for cbField and the options it presents. These options will allow the user to select which field they would like to update/edit.

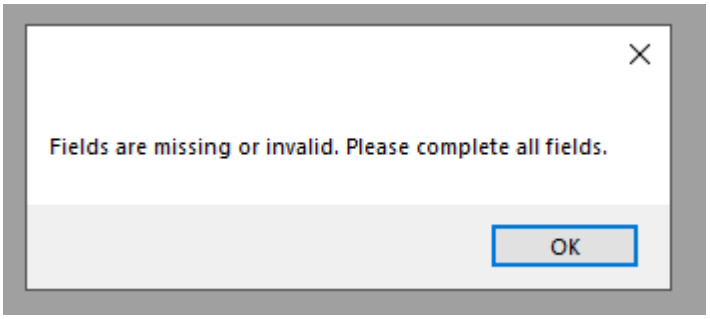


Here we can see the user has attempted to do an update without selecting any records to update

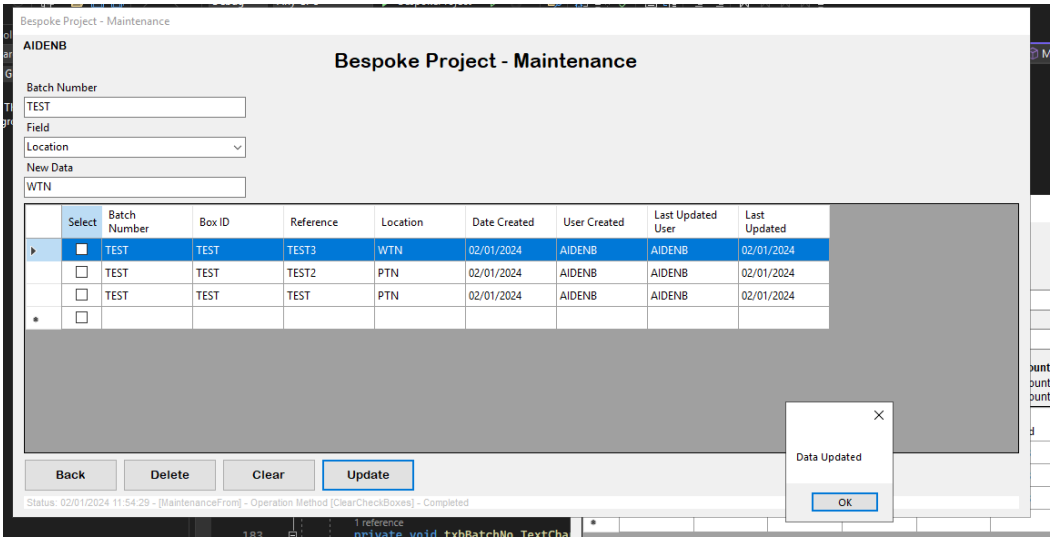
Here an error has popped up because the user has clicked the delete button without entering a batch number



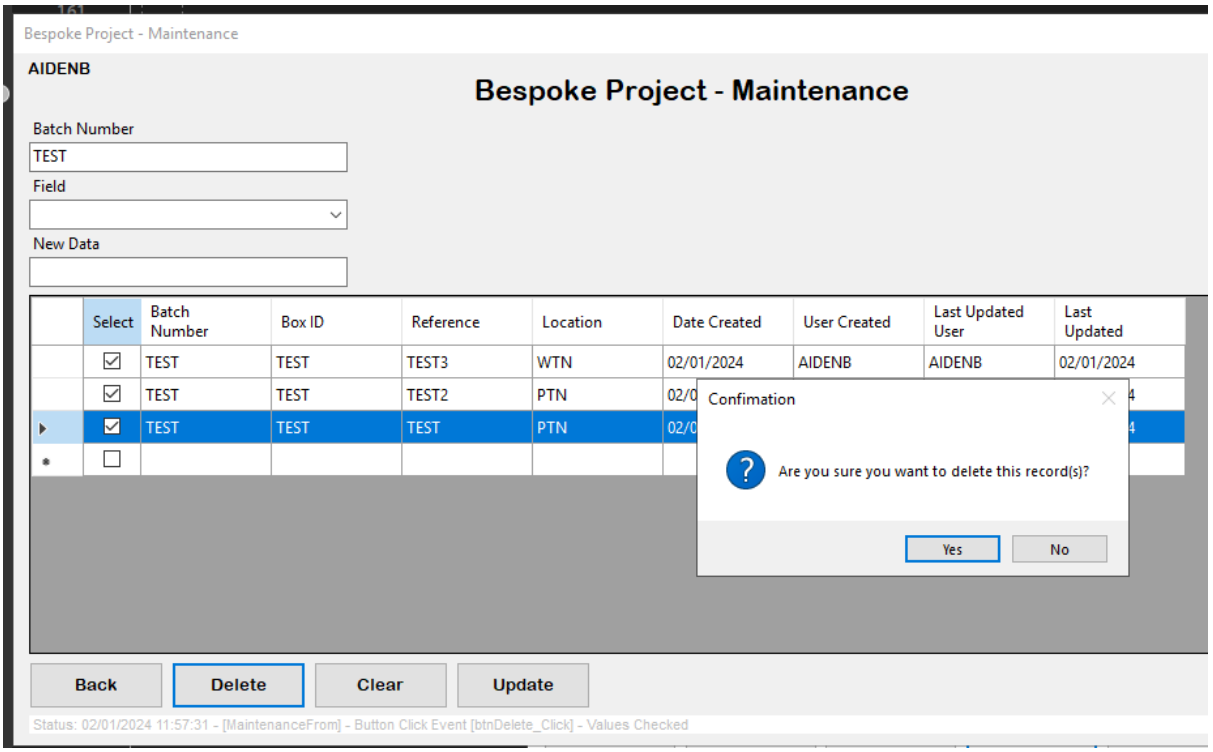
The error here occurs when the user does not complete one or more fields



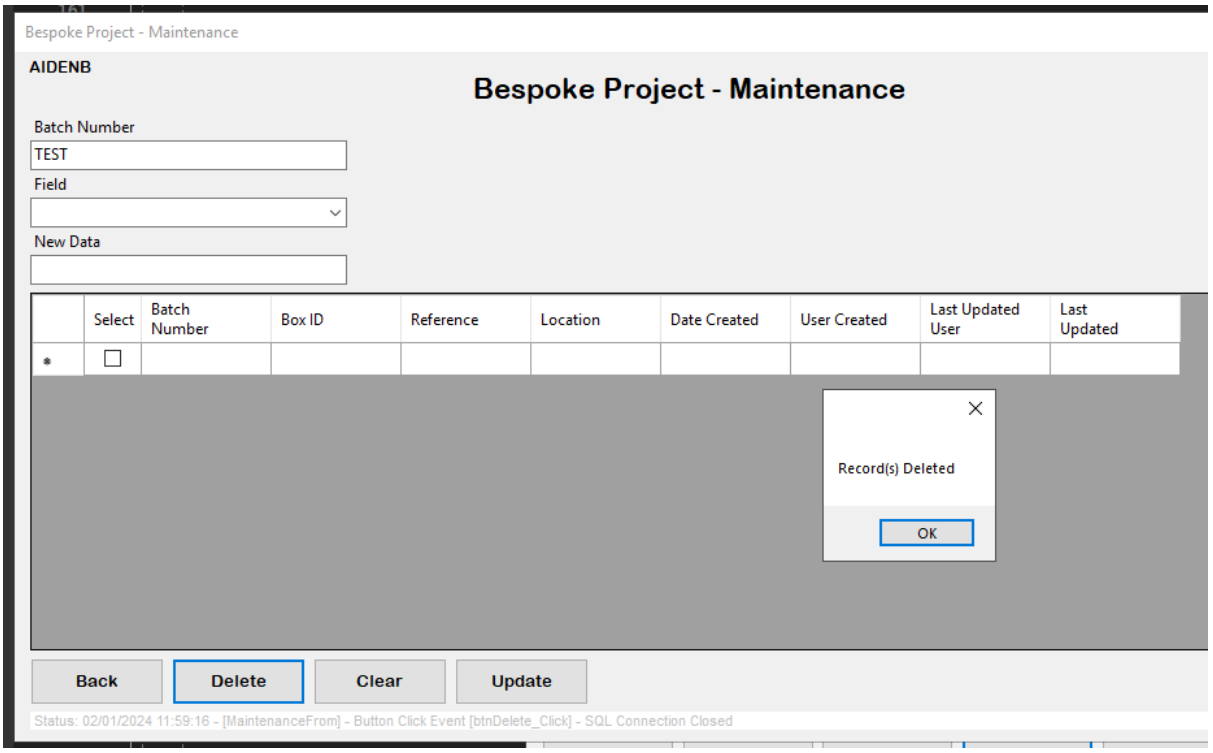
Here we can see a user has updated a record in the database, a message box has appeared informing the user of that the update has happened and the table showing the date is showing the updated record and it's new data, this is to provide visual feedback to the user.



When the user wishes to delete a record, a message box appears asking them if they are sure they would like to do so, this to add an extra layer of input to prevent records from being deleted by mistake. We can see this here.



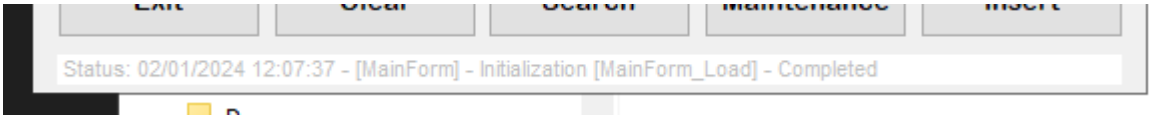
If the user clicks yes to the message box, the record will be deleted and another message box will appear informing the user of this, this is to provide visual feedback to the user.



Logging

During development, to aid in debugging and diagnosing potential issues, a logging process was implemented with two components.

The first part is a status box that displays the last process executed by the application. This status box is visible on both the Main Form and the Maintenance Form



The second part involves a log file, which records the same information displayed in the status box. This process is controlled by a hardcoded boolean parameter. When set to true, it creates a log file (if one doesn't already exist) and continuously populates it with data as the user interacts with the application.

```
public bool Logging = true; // Set Logging Status
```

The logging process is executed through a method, ***private void LogBook(string logStep)***. Every significant step across all application processes and methods invokes this method, filling its parameter with a brief description of the executed step.

Each Logbook entry follows a convention: **Time - Form - Process Category [Method] - Description/Status.** In this format, Time represents the date and time of the entry, Form indicates the form where the entry occurred, Process Category specifies the type of process (Button Click Event, Operational Method, Initialization, etc.), and Description/Status outlines what transpired during that entry

Here we can see the code for the LogBook method

```
73 references
private void LogBook(string logStep)
{
    string filePath = $"C:\\BespokeProjectLog\\Log\\BespokeProject_LogBook_{userName}.txt";
    string logStepString = $"{DateTime.Now} - [MainForm] - {logStep}";
    string newLogBook = $"{DateTime.Now} - NEW LOGBOOK CREATED: {filePath};";

    if (Logging == true)
    {
        if (logStep == "New Login")
        {
            logStepString = "//----------------------------------------------------------------------------------------------------------------------------------//";
        }

        try
        {
            if (File.Exists(filePath))
            {
                File.AppendAllText(filePath, Environment.NewLine + logStepString + ";");
            }
            else
            {
                File.WriteAllText(filePath, newLogBook);
                File.AppendAllText(filePath, Environment.NewLine + logStepString + ";");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error saving Logbook: " + ex.Message);
            Application.Exit();
        }
    }
    lblStatus.Text = $"Status: {logStepString}";
}
```

Here we can see several calls to the ***LogBook*** method, the first of which only occurs when the Try/Catch operation fails, the ***LogBook*** entry will log the failure as well as the exception string given. The second call to the ***LogBook*** shown only occurs when the Try/Catch operation is successful, the entry will log that the SQL Step for the method was executed. The final call to the ***LogBook*** method shown will happen regardless of the result of the Try/Catch Operation and will log that the ***btnInsert_Click*** method has been completed

```
        }

        // Catch Any Errors
        catch (Exception ex)
        {
            MessageBox.Show($"An error occurred: {ex.Message}");
            LogBook($"Button Click Event [btnInsert_Click] - FAILED - Error: {ex.Message}");
        }

        // Set Cursor to Default
        finally
        {
            Cursor.Current = Cursors.Default;
            LogBook("Button Click Event [btnInsert_Click] - SQL Step Executed");
        }
    }
    LogBook("Button Click Event [btnInsert_Click] - Completed");
```

Here we can see an example of a LogBook file created by the application, this example if when the application has first started and no LogBooks currently exist

```
28/12/2023 15:14:53 - NEW LOGBOOK CREATED: C:\BespokeProjectLog\Log\BespokeProject_LogBook_AIDENB.txt;
//-------------------------------------------------------------------------------------------------------------------------------//;
28/12/2023 15:14:53 - [MainForm] - Initialization [MainForm_Load] - Started;
28/12/2023 15:14:53 - [MainForm] - Initialization [MainForm_Load] - User Logged In: AIDENB;
28/12/2023 15:14:53 - [MainForm] - Operation Method [UserCounter] - Started;
28/12/2023 15:14:54 - [MainForm] - Operation Method [UserCounter] - SQL Connection Opened;
28/12/2023 15:14:54 - [MainForm] - Operation Method [UserCounter] - SQL Command Executed;
28/12/2023 15:14:54 - [MainForm] - Operation Method [UserCounter] - SQL Connection Closed;
28/12/2023 15:14:54 - [MainForm] - Operation Method [UserCounter] - Completed;
28/12/2023 15:14:54 - [MainForm] - Initialization [MainForm_Load] - Completed;
28/12/2023 15:21:26 - [MainForm] - Enviroment Event [txbBatchNo_TextChanged] - Triggered;
28/12/2023 15:21:26 - [MainForm] - Operation Method [GetResults] - Started;
28/12/2023 15:21:26 - [MainForm] - Operation Method [GetResults] - DataTable Created;
28/12/2023 15:21:26 - [MainForm] - Operation Method [GetResults] - SQl Connection Opened;
28/12/2023 15:21:26 - [MainForm] - Operation Method [GetResults] - SQL Command Executed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [GetResults] - SQL Connection Closed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [GetResults] - DataTable Populated With Parameters [C], [], [];
28/12/2023 15:21:26 - [MainForm] - Operation Method [GetResults] - Completed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [BatchCounter] - Started;
28/12/2023 15:21:26 - [MainForm] - Operation Method [BatchCounter] - SQL Connection Opened;
28/12/2023 15:21:26 - [MainForm] - Operation Method [BatchCounter] - SQL Command Executed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [BatchCounter] - SQL Connection Closed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [BatchCounter] - Completed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [UserCounter] - Started;
28/12/2023 15:21:26 - [MainForm] - Operation Method [UserCounter] - SQL Connection Opened;
28/12/2023 15:21:26 - [MainForm] - Operation Method [UserCounter] - SQL Command Executed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [UserCounter] - SQL Connection Closed;
28/12/2023 15:21:26 - [MainForm] - Operation Method [UserCounter] - Completed;
```

Bespoke Project - Challenges

Maintenance Form
While developing the maintenance form, I encountered an issue where the application failed to retrieve the correct values for the stored procedure when multiple rows were selected. Instead of capturing all selected rows, the application only fetched data from the last selected row.

Upon investigation, I identified the issue within the foreach loop. To address this, I opted for a different approach: specifically searching for rows where the checkbox in the select column is checked and then extracting the reference from those selected rows.

Before

```
// Create a list to store the selected "Ref_No" values
List<string> selectedRecords = new List<string>();

foreach (DataGridViewRow selectedRow in resultsTable.SelectedRows)
{
    string refNo = selectedRow.Cells["Reference"].Value?.ToString();
    if (!string.IsNullOrEmpty(refNo))
    {
        selectedRecords.Add(refNo);
    }
}
```

After

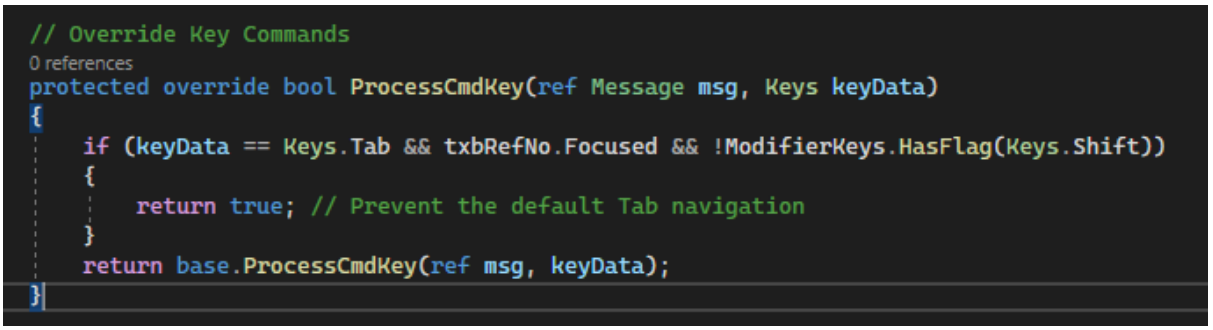
```
// Iterate through all rows in the DataGridView
foreach (DataGridViewRow row in resultsTable.Rows)
{
    // Check if the Checkbox is the Select Column is Checked
    DataGridViewCheckBoxCell checkBoxCell = row.Cells["Select"] as DataGridViewCheckBoxCell;

    if (checkBoxCell != null && Convert.ToBoolean(checkBoxCell.Value))
    {
        string refNo = row.Cells["Reference"].Value?.ToString();

        if (!string.IsNullOrEmpty(refNo))
        {
            selectedRecords.Add(refNo);
        }
    }
}
```

Main Form
An issue arose that prevented me from meeting the criteria outlined in the brief. Specifically, pressing the tab key would trigger the application's insert event while also navigating the cursor. This behaviour hindered the user's ability to seamlessly input reference numbers immediately after entering one.

After several attempts to address this issue, involving methods like overriding default key commands or forcibly focusing on ***txbRefNo***, a resolution was found. By overriding tab navigation only when the tab key was pressed and ***txbRefNo*** was in focus, coupled with a keydown event that triggers the insert event when the tab key is pressed without the shift key, users can now input records by pressing the tab key continuously. Importantly, this allows users to navigate back to ***txbRefNo*** or move to ***txbBoxID*** or ***txbBatchNo*** using Shift+Tab without triggering the insert event.



Bespoke Project - Testing

Testing was conducted throughout the development stages in both real-world and test environments, allowing me to conform to an agile design methodology. An example of this is the integration of counters into the main form. Although not initially specified in the Pre-Development brief, after real-world testing and client review, it was deemed a valuable addition to enhance the user experience.

During testing, more than 70,000 records were inserted into the database using the application. This included both real-world records for potential production use and test records designed to assess the application's limits and capabilities.

In conclusion, the final testing stages of the application have been successfully completed, indicating that the application is now ready for deployment

Bespoke Project - Summary

All criteria outlined in the Pre-Development Brief have been successfully met. The application is designed to streamline the process of receiving and logging bespoke items, especially those with processes and fulfilment differing from standard procedures. Users can efficiently log items/orders, associating them with a batch number and box/bag ID for future reporting and production use. Additionally, the application enables users to search the database for items/orders entered through the system, utilising specified search criteria. Users also have the capability to edit, update, or delete records within the database directly through the application.